We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

4,800

Open access books available

122,000

International authors and editors

135M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities

CLARIVATE ANALYTICS

BOOK CITATION INDEX

INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

4

Character Degradation Model and HMM Word Recognition System for Text Extracted from Maps

Aria Pezeshk and Richard L. Tutwiler
Applied Research Lab, The Pennsylvania State University
USA

1. Introduction

Geographic maps are one of the most abundant and valuable sources of accurate information about various features of bodies of land and water. Due to their importance in applications ranging from city planning and navigation to tracking changes in vegetation cover and coastline erosion, most countries have established dedicated organizations that are responsible for the production and maintenance of maps that cover their entire territories. This has resulted in the production and availability of a tremendous quantity of useful information about every part of the world.

As with other types of documents, new maps are nowadays produced in specialized computer programs, and are easy to manage and update since the individual information layers that form the whole map are available to the map producer. In addition to the change in the method of production, the applications and ways to process maps have also changed in the past few decades. Geographic Information Systems (GIS) have provided new means to analyze, process, visualize, and integrate various forms of geographic data. The input to these systems can be satellite or aerial imagery, remote sensing data (e.g. LIDAR images), raster or vector representations of geographic maps, and any other type of data that are related to locations (e.g. local populations, crime statistics, and textual information). Due to their wide availability, accuracy, and relative cheapness compared to other types of geo-referenced data, geographic maps are probably the most widely used source of information for GIS users. However, the majority of existing geographic maps exist only in printed form. This means that unlike the case for computer generated maps, printed maps cannot be directly used as the input to a GIS since both the end users and the map producers only have access to the dense and complex mixture of regularly intersecting and/or overlapping set of graphical and textual elements rather than the individual features of interest.

Currently the only reliable way of converting printed maps into computer readable format is to have a highly trained operator manually extract the individual sets of features (graphical and textual). The manual feature extraction methods consist of digitization using a digitizing tablet, and heads-up digitizing. In the first method, the paper map is placed on top of the digitizing tablet, and the operator traces over lines and other objects of interest using a stylus or a digitizing puck (a device with crosshairs and multiple buttons that enable data entry operations). In heads-up digitizing (otherwise known as on-screen digitizing) on the other hand, the paper map is first scanned into a digital image. The operator then traces over every single object of interest on the computer screen using a mouse. Since zooming into difficult

sections of the map and editing capabilities are supported in this case, heads-up digitizing is now the more popular method for digitizing the graphical features in paper maps. In addition to the graphical elements, the textual content of maps also needs to be manually extracted and assigned to the appropriate objects in the digital copy. Manual conversion of maps into usable digital format is therefore a tedious, labor intensive, and error prone process. While these procedures may be viable for small scale operations that only involve a few maps, they are unpractical for large scale conversion of whole map repositories. A system that can automate this process is therefore essential for GIS users to replace the manual map conversion methods and provide them with easy access to the vast number of printed maps that exist in various archives and produced by different organizations around the globe.

In this chapter we discuss a custom recognition system for text extracted from maps. This work is part of a larger system that enables automatic sequential extraction of various graphical features, and subsequently processes the remaining image for text grouping, reorientation, and extraction (Pezeshk & Tutwiler, 2008a;b; 2010b). The recognition engine presented here extends the application of the system we developed in (Pezeshk & Tutwiler, 2010a) by enabling multi-font and segmentation-free recognition.

We focus on this particular problem because unlike other parts of the map conversion process, text recognition has received little attention in the research carried out in the field of automatic map understanding systems (as will be discussed in Section 3). Moreover, the level of noise and deformities in text extracted from maps can be too severe for commercial Optical Character Recognition (OCR) systems, resulting in low recognition rates and thereby higher user involvement in correcting the errors in post processing.

The system proposed here can find a variety of applications. Combined with our previous work in (Pezeshk & Tutwiler, 2008a;b; 2010b), we obtain a complete map understanding system that can digitize both the graphical and textual content of printed maps. Since both types of features are extracted at the same time, the text recognized using our system can easily be associated with the other digitized objects in the image (e.g. road labels with the corresponding road segments) in order to obtain an exact digital representation. Moreover, the recognized text can be entered into a database and used to search for maps that contain a particular street or area of interest, or to classify map archives in a similar manner according to their text content. Since geographic maps are accurately geo-referenced, the text can also be used in conflation with satellite or aerial imagery to automatically tag locations or objects that exist in these types of images.

The remainder of this chapter is organized as follows. First, in Section 2 we discuss the challenges encountered in the extraction and recognition of the text content of scanned maps. A brief overview of research in the fields of document image analysis and map conversion algorithms is then presented in Section 3. Details of the different components of our text recognition system which include a custom noise model for the generation of artificial training sets, preprocessing algorithm for automatic text size normalization and orientation correction, and the Hidden Markov Model (HMM) based text recognition engine are subsequently discussed in Sections 4.2, 4.3, and 4.4, respectively. Finally, experimental results are shown in Section 5 before concluding the chapter in Section 6.

2. Challenges

Geographic maps are designed to convey the largest possible amount of information and details regarding both the natural features (e.g. rivers and vegetation cover) and manmade

structures (e.g. roads and major buildings) that are located on any given expanse of land or water. Topographic maps go a step further and incorporate three dimensional topography data which are represented by contour lines. The superposition of the various graphical features and the textual information that is used to label them onto a single two dimensional document results in a complex mixture in which the feature layers (graphical or textual) are regularly intersecting and/or overlapping with each other and thereby no longer individually accessible. Moreover, the common existence of colored or textured backgrounds in maps means that the range of colors that can be used for the various objects in the foreground is limited to only a handful of colors (typically brown, black, blue, and green) in order to maintain their readability. Hence in most cases the same color is used to represent more than one type of feature. While the human visual system is fully capable of discerning each individual type of feature, the density and regular intersections between the different constituent layers of a geographic map comprise the primary challenges in extracting its textual content.

Scanning artifacts such as blurring, aliasing, and mixed color pixels further complicate the text extraction process and its subsequent recognition by affecting the integrity of features in the map image (Khotanzad & Zink, 2003). Geographic maps are particularly susceptible to scanning artifacts since both the graphical features such as roads and contour lines, and the text consist of thin linear segments that are more severely affected by the blending of colors across adjacent pixels and erosion of the edges.

The recognition of the text content of geographic maps has its own set of difficulties. Due to the lack of direct access to the text layer, the graphical features need to be extracted first. As each of these features have distinct characteristics, several processing steps (such as color processing, binarization, and morphological operations) are required to sequentially extract each type of feature until a text only image can be obtained. Misclassification errors that place character segments in any of the graphics layers, or misidentify parts of non-text objects (such as line fragments from the road layer) as belonging to the text layer can result in mild to severe levels of noise and defects in the extracted characters. Furthermore, the text only image will still need to undergo additional processing to group the characters into their respective strings and reorient them to the horizontal direction. Reorientation errors are thus another possible challenge that can cause problems for the segmentation of whole words into individual characters and the recognition of the text.

Text recognition is a well studied subject and is considered a solved problem for the majority of end users who only need to process regular high quality office documents. However, the recognition rates of commercial OCRs are known to drop significantly in the presence of less than ideal conditions (Baird, 2007). Degradations to the quality of the extracted characters that are due to the accumulation of errors from the various processing steps will therefore require the development of a custom text recognition system that is specifically designed and trained according to the anticipated types and levels of defects present in this type of text.

3. Document Image analysis & map conversion systems: A brief overview

The term document image analysis is referred to any set of algorithms and systems that are concerned with the automatic interpretation of printed or handwritten documents. While nearly every new document or drawing is generated by computer software, there is still an immense amount of archived printed material such as engineering drawings, blueprints, microfiches, and maps that need to be converted into machine readable format. Many other

types of documents such as checks and forms that are filled by hand on a regular basis also need to be eventually processed by a computer system so that their content can be transmitted more easily or electronically filed. Document analysis has therefore been an active area of research since the early days of image processing and computer vision.

In forms and checks the graphical elements serve to provide reserved spaces for the text, or act as separators between different parts of these documents. As a result the graphical objects (such as rectangular enclosures, straight lines, and logos) which do not provide any useful information can be simply detected and removed using common line detection algorithms (e.g. Hough transform) or basic morphological operations, and the main steps in the processing of such documents will typically consist of skew correction (so that the extracted strings are correctly reoriented to the horizontal direction) and handwritten text recognition (Hull, 1998), (Sarfraz et al., 2005), (Guillevic & Suen, 1997), (Namane et al., 2006).

Engineering drawings and blueprints contain a large amount of information presented as both text and graphics. Therefore both types of features need to be extracted with high accuracy and the tolerance level for misclassification errors is low. However, text and graphics rarely intersect with each other, and the two classes of features can be typically separated from each other using connected component analysis (graphical features tend to be larger in pixel area than characters). Smaller graphical features in such documents mostly consist of arrows, dashed lines, and hatched areas. As the general shapes and characteristics of such objects are known beforehand, specialized modules can be subsequently applied to the image to obtain a text only image. Therefore the separation of the graphical elements has little effect on the integrity of the extracted text layer, and both commercial and custom OCR systems can be used for its recognition. These conditions have led to the development of many practical digitization systems (Tombre et al., 2002), (Dori & Wenyin, 1999), (Lu, 1998), (Wenyin et al., 2007), (Chen et al., 1999).

Due to the difficulties associated with the extraction of features from geographic maps in general, and topographic maps in particular, many of the systems presented in the literature have focused on simpler maps where features are well separated from each other or printed in colors that facilitate the map digitization process (e.g. see (Pouderoux et al., 2007), (Roy et al., 2007)). Other systems mostly consider the extraction of one or more types of features from complex maps ((Chiang et al., 2005; Dhar & Chanda, 2006; Gamba & Mecocci, 1999; Kerle & Leeuw, 2009; Khotanzad & Zink, 2003; Leyk et al., 2006)), but do not address the problems of non-horizontal text or character recognition.

Cao et al. (Cao & Tan, 2002) used a commercial OCR to recognize text extracted from street maps that have limited intersection between different features. Li et al. (Li et al., 2000) rely on a cooperative method to extract graphical features from complex maps. An OCR that uses template matching and trained using actual character prototypes that are manually labeled is then used to recognize the text. Velázquez and Levachkine (Velázquez & Levachkine, 2003) assume most of the graphical features can be removed using binarization of the original image. A method called V-lines is subsequently used to separate the text in the remaining image from possible attached artifacts or symbols. The extracted text is finally recognized using an artificial neural network in conjunction with a gazetteer. Similar recognition systems have been reported in (Levachkine et al., 2002) and (Gelbukh et al., 2003).

In the next section we discuss our proposed approach that aims to minimize the required amount user involvement in the extraction of the text content and overcome the challenges encountered in character recognition for topographic maps.

4. Proposed system

4.1 Text extraction

The text extraction procedure involves the sequential application of several modules that perform text/graphics separation, followed by text grouping and reorientation as shown in Figure 1. The particular set of procedures discussed here are primarily targeted at topographic maps produced by the United States Geological Survey (USGS) since they encompass many of the difficulties discussed in Section 2 such as variety and density of features, and usage of the same color for several of the intersecting features. The different components of the text extraction system are designed to contain easily adjustable parameters, and to have minimal reliance on heuristics and prior knowledge about the underlying map image. The extension of the proposed algorithms to maps produced by other organizations can then be easily achieved by proper selection of these parameters. Here we assume that the individual binary word images that constitute the input to our recognition system are already extracted using this multi step process. We therefore provide only a high level description of its different elements below and refer the interested reader to the related publications for details of each algorithm:

- *Contour Line Extraction*: Contour lines are first extracted based on their color property. However, due to the mixed color pixel problem (as mentioned in Section 2) the raw colors in the original image cannot be used. Instead, we use the algorithm in (Pezeshk & Tutwiler, 2008a) to create a false color image in which the intra-cluster variance of colors of contour line pixels has been reduced. A sample patch containing only contour line pixels is then manually selected, and a distance threshold computed according to the statistics of the sample points is used to extract the contour lines. A similar procedure can be repeated for any other type of feature that is printed in color (such as rivers). The remaining image consisting only of features printed in black (such as roads, text, and man made structures) is subsequently binarized.

- *Linear Feature Extraction*: Roads and boundary lines only gradually change directions, and can be considered to be piecewise straight or nearly straight. The algorithm presented in (Pezeshk & Tutwiler, 2010b) extracts such linear features based on this observation. Analogous to the process of describing a vector in terms of a number of basis vectors, the input binary image is dissected into multiple directional feature planes which contain the projections of the edges of every object towards a number of primary directions. Due to the aforementioned "straightness" property, every line will have one or more projections in the directional planes that are longer than edge projections from character segments and other non-line objects. A set of directional morphological operations that operate on non-isotropic neighborhoods are subsequently used to automatically eliminate these shorter edge projections, and to reconstruct whole lines from the remaining core edge segments. The linear feature layer is then defined as the union of the restored line segments from all the directional planes, and extracted from the image. This directional line extraction method has the benefit of being capable of extracting lines even when they are intersecting with the text layer, while causing minimal damage to the integrity of the characters.

- *Clutter Removal*: The remaining non text objects consisting of buildings, dashed lines, and line fragments are removed using morphological operations, a dashed line detection algorithm, and a search process that identifies isolated linear artifacts, respectively (Pezeshk & Tutwiler, 2008b; 2010b).

- *Text Grouping & Reorientation*: Grouping of characters into their respective strings is performed using pyramid decomposition with Gaussian kernels. Going down through the consecutive levels of the pyramid, the successive downsampling and smoothing cause adjacent characters to merge together. When the image is upsampled to its original size and binarized, the general area of each string can be identified by the blobs formed in the upsampled image. Individual strings can then be reoriented to the horizontal direction according to the orientation of the longer edge of the minimum area enclosing rectangle fitted onto each of the blobs (Pezeshk & Tutwiler, 2010b). Examples of the extracted text are shown in Figure 2.

4.2 Noise model

Text recognition systems can in general be divided into two categories. Those in the first group rely on a segmentation algorithm (see e.g. (Casey & Lecolinet, 1996)) to obtain the individual characters in every word image, and then send each of the extracted characters to a classifier trained on single character images for recognition. The performance of such systems is mainly determined by the accuracy of the segmentation procedure. As seen in Figure 2, words extracted from maps may contain broken and/or conjoined characters, and italicized typeface, which are all conditions that lead to a higher likelihood for segmentation errors. The cascading effect of such errors coupled with the difficulties in recognizing characters with mild to severe levels of noise and defects will consequently result in unacceptable overall performance rates.

One of the most promising approaches for the recognition of heavily noisy or degraded characters belongs to the second general category of recognition systems that do not rely on prior segmentation of a word image into its constituent characters. Kuo et al. (Kuo & Agazzi, 1994) and Aggazi et al. (Agazzi et al., 1993) obtained excellent recognition results for severely degraded word images using pseudo two dimensional HMMs that modeled all or major parts of words which were known to belong to a small sized lexicon. The training procedure for these systems would therefore also consist of images of whole or major parts of the known words. Since the text in maps mainly consists of names that may not be known a priori, using such an approach is not possible in this case.

Here we use a procedure that relies on implicit segmentation of the word images into individual characters, followed by recognition using a combination of classifiers. Similar to the segment-then-recognize systems, the training data for our recognition engine will therefore consist of single character images.

Large scale experiments have shown that the dominant factors in determining the performance of text recognition systems are the size and representativeness of their training data (Baird, 2007). The ideal set of training images in our application would therefore consist of actual single character images that have been extracted from real maps. However, geographic maps only contain a small amount of text that is difficult to extract and label, and appears in multiple different fonts. The collection of enough number of real character samples for training would therefore require processing many different map sheets, which in addition to being very difficult and time consuming, may not be possible if a large number of maps with similar features are not available. We therefore propose using a custom noise model that replicates the types of defects and deformations seen in actual characters extracted from maps, and allows us to generate training sets with arbitrary size.

Fig. 1. Performance evaluation of the text and graphics separation modules; (a) Original map image, (b) Extracted Contour Lines, (c) Foreground image, (d) Extracted linear features , (e) Text image, (f) Grouped characters and their respective minimum area enclosing rectangles
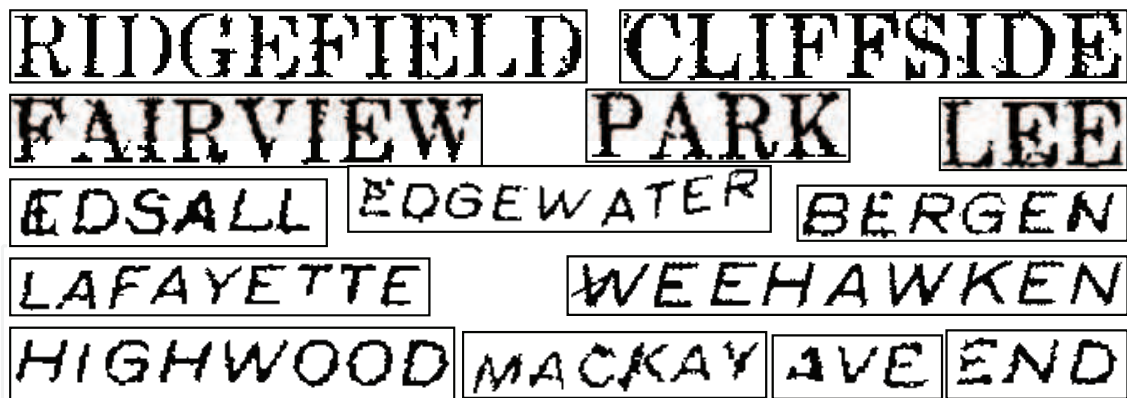
Fig. 2. Samples of extracted words

We use the degradation model introduced by Baird (Baird, 1992) as the starting point of our defect model. Instead of the full model, here we pick the following four types of degradations that were deemed to be most relevant to our application:

- *Spatial Quantization*: A degradation that is a function of character size (in units of points) and scanning resolution (in pixels/inch).

- *Blurring*: Deformations caused by the combined effect of the point spread functions of the printing and scanning processes are modeled by the parameter *blur* (in units of output pixels) which is defined as the standard error of a circularly symmetric 2D Gaussian filter.

- *Speckle Noise*: Per-pixel additive noise is modeled by the parameter *sens* (in units of intensity) which is defined as the standard error of a normal distribution with zero mean. This parameter models the side effects of the color transformation used in the contour line removal step.

- *Thresholding*: Effects of binarization of the map image are modeled using the parameter *thresh* (in units of intensity, assuming white and black are represented by 0 and 1 respectively).

To generate an artificial character image using this model, which we refer to as the Truncated Defect Model (TDM), a noise free binary image of the desired character is first rendered at around ten times the output resolution. This image is then shifted randomly within [0,1] (in output pixel units) in the horizontal and vertical directions, blurred, and subsampled to output resolution. Per pixel speckle noise is subsequently added to this image, and the whole image is binarized using the parameter *thresh*.

Another type of artifact that can be found Figure 2 are short linear attachments that were originally parts of intersecting lines. The model above is thus supplemented by a random linear artifact generator that replicates this type of artifact using an iterative procedure, and allows for complete randomization and parametrization of the direction, connectivity, point of origin, length, and width of line attachments for character images. The details of this model which we refer to as the Extended Defect Model (EDM) are as follows:

Step 1) *Initialization*: Assuming uniform distribution over the range of possible values for each parameter, the defining attributes of the linear artifact are randomly selected during initialization. First, an initial growth point G_0 is randomly selected from the pixels on the perimeter of a character image generated by the TDM. Due to the relatively short length of the linear artifacts, they can be assumed to be straight or nearly straight. Hence
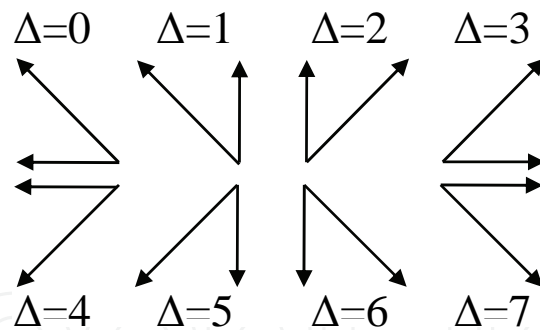
Fig. 3. Possible growth directions from a previous pixel

the second random selection picks one of the eight combined directions in Fig. 3 as the general direction of growth for all subsequent iterations. Each direction Δ identifies two candidate directions of growth and therefore two new neighboring pixels at each iteration. The selection of one of these two candidate growth pixels is performed based on the probability pair $(p, 1 - p)$ in each iteration, where the value of p is picked randomly during initialization from $V_p = [0.1, 0.2, \ldots, 0.8, 0.9]$. Finally, the parameter L_{seg} is randomly selected from the vector $V_L = [2, 4, \ldots, L_{max}]$ to determine the total length (in number of pixels) of the generated line segment.

Step 2) *Growth*: At each iteration t, one of the two neighboring pixels N_1 and N_2 of the previous growth location G_{t-1} is randomly selected based on the probability pair $(p, 1 - p)$ as the new growth pixel G_t. Discontinuities along the intersecting line fragments are then replicated by using a second random process to determine whether G_t will be an ON (or OFF) pixel (in other words 1 or 0, respectively) with probability q (or $1 - q$). The direction orthogonal to current direction of growth at each iteration is along the line's width. Linear artifacts that are more than one pixel thick can therefore be obtained using the same general procedure, but at each iteration one or more pixels should be added (grown) along the width of the line.

Step 3) *Stopping criteria*: Continue until $t = L_{seg}$

4.3 Text normalization

The feature vectors that are used as the input to the recognition engine should all have the same length. Every extracted word image thus needs to be normalized to a fixed height according to the desired length of feature vectors. Under normal circumstances, the height of individual characters are accurate indicators of the actual font size, and thereby the normalization factor. However, due to the presence of attached artifacts that may extend above or below the actual boundaries of characters, noise and defects, and possible deviations in orientation from the horizontal, the actual character size cannot be inferred based on the heights of characters here. Instead, we use a custom preprocessing algorithm to normalize words printed in either lower case or upper case to the appropriate target size.

Words printed in upper case can be considered to be encased between two imaginary lines that pass through the tops and bottoms of its constituent characters. Estimation of these lines, named cap line and baseline, respectively, should therefore enable us to find the actual bounds of characters and thereby correctly normalize each and every whole word image to the desired size.

Similar to any other regression problem, we first need to select a source of data, and an estimation algorithm. Since the word images are horizontal or nearly horizontal, we can use

the pixels located on the outer contours of characters as viewed from the bottom, which we refer to as the bottom profile, to estimate the baseline. However, due to the presence of noise and attached artifacts, and shapes of characters such as H, F, and M which contribute many points that deviate significantly from the baseline, this set of points is heavily contaminated with outliers.

Among the various parameter estimation techniques, the RAndom Sample Consensus (RANSAC) algorithm (Fischler & Bolles, 1981) has been shown to perform robustly even in cases where the total population of data consists of up to 50% outliers. We therefore use RANSAC to estimate the baseline from the bottom profile of each word image. The cap line can be estimated in a similar manner from the top profile of every image, which is defined as the outer contour of characters as viewed from the top. A beneficial side effect of this procedure is that once the baseline and cap line are estimated, any artifacts that extend below or above these lines can be automatically eliminated in order to reduce the overall level of noise in the word image (if the extension in the bottom of the letter Q passes below the baseline in the font being used, the baseline will still be correctly estimated, but points below the baseline should not be eliminated). Furthermore, the average of the slopes of the baseline and the cap line can be used to correct slight orientation errors that may exist in a word image. Finally, the vertical distance between the two enclosing lines is used to compute the normalization factor and resize the word images to the target size. An example showing the different steps of this procedure can be found in Figure 4.

While the cap line is not defined for characters printed in lower case, the general procedure discussed above can still be used to normalize this type of text. However, this time we use the baseline and the mean line, which is an imaginary line that passes through the tops of characters such as "a" and "c", to normalize the word images. The estimation of the two enclosing lines is more difficult in this case compared to upper case characters since many more actual character segments, namely the ascenders (parts of characters such as "t" and "f" that rise above the mean line) and descenders (parts of characters such as "p" or "g" that extend below the baseline), contaminate the top and bottom profiles, respectively, and increase the ratio of outliers in the estimation data.

The majority of pixels in words printed in lower case can be considered to be located between the baseline and the mean line. We can therefore find this general area by computing the horizontal histogram of pixels in a word image, and eliminate most of the ascender and descender pixels by removing any rows in the image that contain less than 50% of the maximum count in the histogram. The baseline and the mean line will then be estimated from the top and bottom profiles of this new image, as shown in Figure 5. Similar to the procedure for capital letters, the average of the slopes of the two lines can be used to correct slight orientation errors that may exist in a word image.

4.4 Recognition engine

Hidden Markov models have had a long and successful history in the field of speech recognition. The similarity between many of the problems encountered in this field and those in text recognition has led to the widespread use of HMMs in both printed and handwritten character recognition, and in particular noisy, degraded, and/or connected text recognition (Bose & Kuo, 1994; Kim et al., 1997; Koerich et al., 2000; Likforman-Sulem & Sigelle, 2009; Schenkel & Jabri, 1998). The system described here also uses an HMM based classifier, and is most similar to the general structure of the HMM based recognition system proposed by Elms et al (Elms et al., 1998). However, the two systems are different in the type of
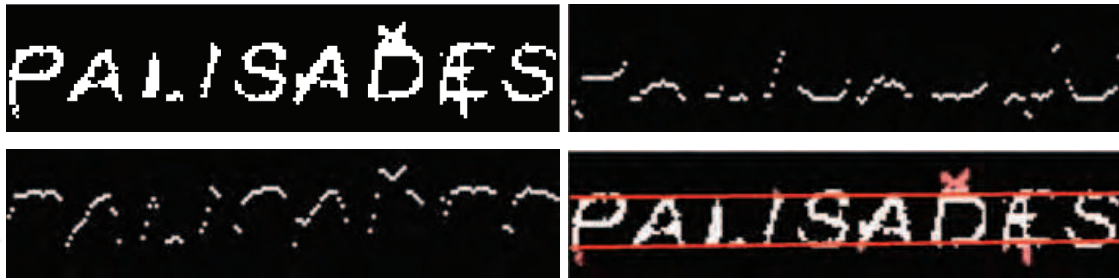
Fig. 4. Preprocessing using RANSAC for words printed in capital letters; Original image shown in the top left, top and bottom profiles shown in the top right and bottom left, and the fitted baseline, capline, and detected artifacts are shown in the bottom right
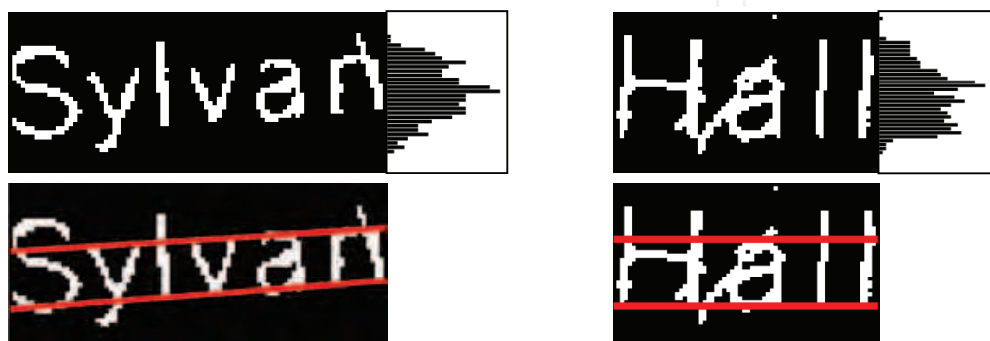

Fig. 5. Preprocessing using RANSAC for words printed in upper case and lower case; from top to bottom: Original word image and its corresponding horizontal histogram, Fitted baseline and mean line

feature vectors used for classification, clustering algorithm employed in the generation of codebooks, modeling of the white space in between characters, and the incorporation of letter transition probabilities into the Level Building Algorithm (LBA) (Rabiner & Levinson, 1985) that performs the implicit segmentation of word images into individual characters. Moreover, the system in (Elms et al., 1998) is trained on real character images and includes contextual knowledge in the form of a lexicon, whereas here we use artificial character images (as discussed in Section 4.2) for training and only use a simple language structure (letter transition probabilities) to improve the performance of our system. The different components of the proposed recognition engine are described next.

4.4.1 HMM structure & training
The full structure of a 2D image can only be properly captured if the relationships between the different points in both dimensions of the image are simultaneously captured. This method of feature extraction leads to the two dimensional (or planar) HMMs as seen in (Levin & Pieraccini, 1992), or the pseudo two dimensional HMMs in (Kuo & Agazzi, 1994). Alternatively, the information in each dimension can be modeled independently by separate HMMs so that simpler one dimensional HMMs can be employed. Here we have taken the latter approach, and use the consecutive rows and columns of normalized character images as features. Accordingly, two HMMs will be trained for each character class based on the quantized horizontal (row wise) or vertical (column wise) features extracted from images of characters in their corresponding classes. During recognition, the likelihood values for the horizontal and vertical HMMs will be combined based on the procedure explained in
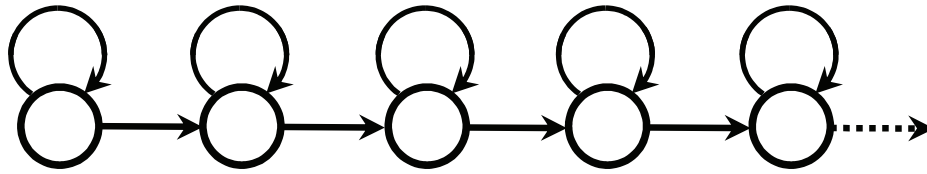
Fig. 6. Left to Right HMM with 5 states

Section 4.4.3. Quantization of the feature vectors is implemented according to two separate codebooks. The *K-means* algorithm is used to generate one codebook based on the horizontal feature vectors extracted from the training images belonging to every character class, and another one in a similar fashion from the vertical features (here we chose 45 cluster centers for both cases based on experimental results). All row wise and column wise features are then quantized using the nearest neighbor rule on their corresponding codebook.

We use the simple left-right structure shown in Figure 6 for each HMM. In this structure, transitions are restricted from a current state S_i only to the immediate next state S_{i+1} or back to itself so that each fixed number of observations will be generally modeled by one state. The total number of states is therefore directly dependent on the length of the observations and thereby the normalized size of the character images. Here we chose the parameters of the TDM to create artificial characters from the set A:Z using different fonts with a uniform height of 28 pixels, and centered each noisy character within a 38 by 38 pixel image (the padding on the sides of each character provides the necessary space for variations between width of characters such as "I" ad "W", and the linear artifacts generated by the EDM). The number of states for each HMM (horizontal and vertical) was then fixed at 10 states for every character class, so that there are approximately four observations per each state (in reality, some states may represent more or less number of observations depending on the particular character class and levels of noise and defects in the training images). It should also be noted that the prior state probability matrix of each HMM is normally estimated during training. However, the LBA requires the prior probability of the first state of every HMM to be equal to one (Rabiner & Levinson, 1985). Before training, the initial guess for the prior probabilities of the states of each HMM are therefore chosen according to this requirement. These settings will not be affected by the training procedure since parameters that are set to zero will remain unchanged.

4.4.2 Level building algorithm

As discussed in Section 4.2, segmentation of words extracted from maps using explicit methods is not a viable approach. The LBA provides a suitable alternative by simultaneously finding the jointly optimal segmentation and recognition results for each word image. The LBA is essentially an extension of the Viterbi algorithm. While an unknown sequence of observations extracted from a single character image can be matched against an HMM using the Viterbi algorithm, the LBA can use multiple HMMs to split an observation sequence extracted from a whole word image into subsequences and match them to individual models. Given a set of character models $\lambda = (\lambda^1, \lambda^2, \ldots, \lambda^N)$ and an observation sequence $O = (O^1, O^2, \ldots, O^T)$, the algorithm starts at level 1 by progressively scanning through O from left to right. The incremental likelihood of matching the partial sequence $(O^1, O^2, \ldots, O^t)$ to a model λ^n is then computed at each frame t and for every model, and stored in the array $P(1, t, n)$. At the end of level 1, the model that best describes the partial sequence $(O^1, O^2, \ldots, O^t)$ at each frame t and its corresponding likelihood are stored in the arrays $\hat{W}$

and $\hat{P}$:

$$\hat{W}(1,t) = \underset{n}{\operatorname{argmax}} P(1,t,n) \tag{1}$$

$$\hat{P}(1,t) = \underset{n}{\max} P(1,t,n) \tag{2}$$

The computation at higher levels is carried out in a similar fashion, except that the cumulative likelihood for each model and at each frame t picks up from the best likelihood at frame $t-1$ in the previous level ($\hat{P}(l-1,t-1)$). In addition, a backpointer is maintained to keep track of the ending frame of the best previous model. This procedure is continued until a maximum number of levels l_{max} have been reached, at which point the algorithm backtracks from the best model at the last time frame T ($\hat{W}(l_{max},T)$) using the previously constructed backpointer array. The jointly optimal recognition and segmentation results are subsequently found from the sequence of best matching models and the backpointer array, respectively.

Horizontal features cannot be extracted from the word image unless the extents of its individual characters are known beforehand. We can therefore only use the vertical (column wise) features to construct an observation sequence for each word image. Accordingly, every word image is normalized in height such that it has a height of 28 pixels and centered within a 38 pixel high window (similar to the height format of the training images), while the width of each image is adjusted to maintain its aspect ratio. The column wise features are then quantized according to the codebook for vertical features obtained in training, and the LBA is applied to the resulting observation sequence based on the set of HMMs trained on vertical features. It should also be noted that in many cases the characters are separated by empty inter character space. A separate HMM is typically used to model this empty space (e.g. see (Elms et al., 1998)), in which case the maximum number of levels of the LBA needs to be increased to account for its occurrence. However, the training images used here already model this empty space by having extra padding to the sides. Since the extra empty space HMM is not needed here, we can use the width of each word image to hypothesize the maximum number of letters and thereby the maximum number of levels that need to be computed in the LBA as follows:

$$l_{max} = \lfloor T/\mu_{char} \rfloor + C \tag{3}$$

In this equation T is the total width of the word image (equivalent to its number of columns), μ_{char} is the mean character width found from the training images, and C is a constant which accounts for inter character blank spaces and the much smaller width of the letter "I" (here we set $C = 3$). Each normalized word image should therefore be processed by the LBA for levels 1 through l_{max}. The level l that generates the highest overall likelihood $\hat{P}(l,T)$ will then determine the optimal number of letters and the corresponding segmentation points for that word image.

In its current form, the LBA might match an unreasonably short or long sequence of observations to a single character model (for instance, the model for the character "A" may be matched against a sequence with 10 or 100 observations). Assuming that the width (or duration) of each character can be modeled by an individual Gaussian distribution, a temporal constraint can be included in the LBA by modifying the computation of the cumulative likelihoods $P(l,t,n)$ as follows (Rabiner & Levinson, 1985):

$$\tilde{P}(l,t,n) = P(l,t,n) \cdot P_n(d_{\Delta t})^{\alpha} \tag{4}$$

In this equation P_n is the Gaussian distribution for the width of character model λ^n with mean μ_n and variance σ_n, $d_{\Delta t}$ is the duration of the current observation subsequence being matched to model λ^n, and α is a weighting factor. The parameters for each individual Gaussian distribution (μ_n and σ_n) are obtained from the training images of their corresponding character model, and the value for α is optimized experimentally.

The absence of a syntax constraint in the LBA means that any character model is equally likely to follow another character model from a previous level. However, statistical analysis of the words in any given language shows that some pairs of characters are more likely to appear than others. For instance, in English the letter "Q" is always followed by the letter "U", and therefore the LBA should penalize the matching of letters other than "U" in the computation of cumulative likelihoods if the best character model at the previous level is a "Q". A bigram model specifies the probability with which a letter follows a previous letter in a word for a specific language. This simple statistical language model is consequently incorporated in the LBA using the following probability array in order to further improve its performance:

$$\check{P}_{n|m}(l-1, t-1) = \hat{P}(l-1, t-1) \cdot P_{tr}(n \mid m)^\beta \tag{5}$$

where n is the character corresponding to the current model λ^n being matched to a partial sequence of observations at level l, m is the best character from the previous level ($\hat{W}(l-1, t-1)$), $P_{tr}(n \mid m)$ is the appropriate letter transition probability, and β is weighting factor. At the beginning of each new level, $\check{P}_{n|m}(l-1, t-1)$ will thus substitute $\hat{P}(l-1, t-1)$ in the computation of cumulative likelihoods for each new character model λ^n to include the bigram model. Here we obtained the letter transition probabilities from Konheim (1981), and the value for β was optimized experimentally.

4.4.3 Combining classifiers

The segmented characters obtained from the LBA are processed by a separate classifier that also uses the horizontal HMMs for more robust recognition results. Each character is first normalized in size and centered within a 38 by 38 pixel image (similar to the training images) and its horizontal and vertical features are quantized according to their corresponding codebooks. Two arrays of likelihood values can now be computed for each character by matching the column wise and row wise observation sequences against all HMMs trained on vertical and horizontal features, respectively. A third array of likelihood values is obtained from evaluating the width of each character in the Gaussian width models of every character class. Assuming independence, these three arrays can then be combined into a single likelihood array through point-wise multiplication. However, the likelihood values from the character width models will be weighted similar to (4) using a factor α.

The arrays of likelihood values for each of the characters extracted from a word image can be assembled into a trellis where the number of nodes at each time step corresponds to the number of character classes and the number of time steps matches the total number of letters in that word (equal to the optimal number of levels in the LBA). At each time step (letter in the word), we then assign a score to the nth node (character class) that is equal to the value of the nth element in the combined likelihood array of the corresponding character image. The edges in the trellis represent a transition from a previous character class to the next one. We therefore associate a cost to each transition according the letter transition probabilities (the bigram model). Similar to (5), this cost will be weighted by a factor β.

Each path through this trellis represents a possible sequence of characters matching the word image, and the cost associated with its traversal can be computed based on the scores of the nodes and edges along the individual path. The optimal sequence of matching characters for each word image therefore corresponds to the best path through the trellis, which can be efficiently computed using the Viterbi algorithm.

5. Experimental results

Two experiments were performed to evaluate the effectiveness of the proposed text recognition system on street labels and place names extracted from several USGS topographic maps. In the first experiment, the character images for the training data were generated using the same fonts as the extracted text according to USGS style sheets. In the maps being tested here the street labels were printed in *Univers 53 Extended Oblique* (a sans serif font), whereas the place names used *Century Expanded* (a serif font). Both types of text were printed in all capital letters. In the second experiment, we used the generic fonts *Arial* (sans serif) and *Times New Roman* (serif) for the training data in order to assess the performance of the system in cases where the specific fonts used on the maps are not known. Both experiments were carried out according to the same procedure, and the implementation details that follow should be considered to apply to both cases.

5.1 Training

The artificial character images generated by the noise models described in Section 4.2 should closely mimic the defects and degradations seen in real characters extracted from maps. Correct estimation of the parameters that govern the behavior of these models is therefore critical to the overall performance of our recognition system.

Barney Smith (Smith, 2001) has proposed a method to estimate the blurring and threshold parameters from corners of characters in binarized scanned images, but the estimation of the remaining parameters in Baird's defect model is not discussed. Kanungo et al. (Kanungo et al., 2000) have developed a statistical bootstrapping method which tests the hypothesis that two sets of images belong to the same underlying distribution. Baird (Baird, 1999) showed that this method can be effectively used to make fine and accurate discriminations with regards to the parameters used in his character defect model.

Here we use a simple approach to find the suitable ranges of values for the parameters used in the TDM. Five samples of actual characters for each of the letters M, S, A, N, and R were first extracted from maps (a total of 25 samples). Each of the parameters in the TDM were then allowed to range within specific intervals, and a set of 60 artificial character images were rendered using each of the parameter settings. The Mean Square Error (MSE) of the images in each set of artificial characters with respect to the real character samples was subsequently computed, and averaged over the number of images in each set (60) and the total number of actual letter samples (25). A few combinations of parameter settings that produced the smallest MSEs were then selected to create the training images. Figure 7 shows samples of actual characters extracted from maps, and the artificial characters generated by the TDM and EDM using the above procedure.

For each character class A through Z and each font, 240 artificial characters were created using the TDM. The images for each character class were then combined such that characters from each font appeared in alternate positions in the training data. A second set of artificial characters were created and used as the input to the EDM (a grand total of $4 \times 26 \times 240 \approx$
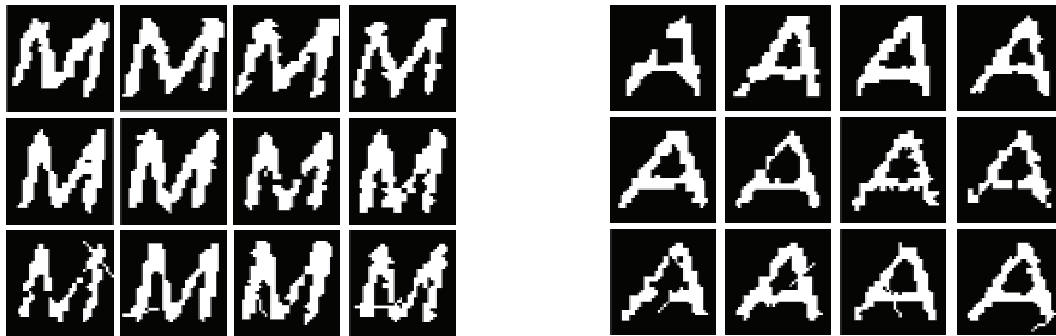
Fig. 7. Samples of characters M and A when the fonts are known; First row are actual character samples from maps; second and third rows of images generated using the TDM and EDM, respectively

25000 character images). The HMMs were trained in two stages. First, the general structure of each character class was learned using only the images generated by the TDM. The noisier EDM images were then used to fine tune the parameters in each HMM.

5.2 Exception handling

Curvilinear text comprises only a very small percentage of the total amount of text in maps. Since individual characters in such words are situated along a curve (rather than a straight line), the preprocessing algorithm described in Section 4.3 cannot be used in such rare cases. As the LBA relies on proper normalization of the input words, a different character segmentation procedure should be used here.

During pyramid decomposition and the subsequent fitting of the individual word blobs with the minimum area enclosing rectangle (Section 4.1), curved words will show high variation in the distance between the bottom (or top) side of their enclosing rectangle and the bottom of the blob, whereas flat words will be snugly fit with the enclosing rectangle. Curved words can therefore be automatically identified.

The individual characters in curvilinear text are better separated from each other compared to regular text, which means they are better candidates for explicit segmentation algorithms. Here we used the algorithm in (Liang et al., 1994) to obtain the individual characters in such words. The combined classifier in Section 4.4.3 was subsequently utilized to recognize each word image.

5.3 Results

Table 1 summarizes the recognition results for the extracted street labels and place names using HMMs trained on actual and generic fonts (experiments I and II, respectively), and ABBYY FineReader 10 (a leading commercial OCR) which was used to benchmark our system. Here we used the single character edit distance which counts each insertion, deletion, and substitution as one error to compute the recognition rates. As expected, the commercial OCR performed well in cases where the level of noise and defects was low (e.g. the words "HIGHWOOD" and "LAFAYETTE" in Figure 2), but showed a drop in its recognition rate for words that were too noisy or not perfectly horizontal (e.g. the words "EDGEWATER", "MACKAY", and "AVE" in Figure 2). It should also be noted that both systems performed at close to 100% for the words printed in the serif font (*Century Expanded*). This could be an indication that serif fonts have more distinguishing features compared to the sans serif fonts.

Recognition Engine	Correct	Wrong	Recognition Rate
Experiment I (actual fonts)	964	63	93.87%
Experiment II (generic fonts)	936	91	91.14%
ABBYY FineReader10	921	106	89.68%

Table 1. Performance evaluation of recognition of characters: Our system vs. ABBYY FineReader10

6. Summary

In this chapter, we described a custom multi-font recognition system for text extracted from maps. The different components of this system have been designed to minimize the required amount of user involvement so that the overall cost of digitizing paper maps can be reduced. Using only artificially generated training data, this system was shown to produce acceptable recognition rates both when the actual fonts used for the text are known, and when only generic fonts are available. Since this system does not rely on prior knowledge in the form of gazetteers or lexicons, it can be easily configured to work with maps produced by different organizations and printed in languages other than English. However, if such information becomes available the existing system can be modified to take advantage of it using the Dictionary Viterbi Algorithm (Hull et al., 1983).

While the performance of the text extraction and recognition system described here is promising, the fully automatic conversion of paper maps into computer readable format remains an open problem due to the wide variability of the characteristics and features of maps produced by different organizations. As a result a realistic goal for any map conversion system should be to take advantage of user interaction when needed, but to limit the amount of user involvement to only a supervisory capacity.

The current recognition system can still be improved such that words that contain a mixture of both upper case and lower case characters, or letters and numbers can be processed at the same performance rate as words that contain only capital letters. A first step towards that direction would be to change the bigram probabilities so that they will allow transitions between lower case and upper case letters, and numbers, according to probability distributions learned from processing a large enough population of actual text samples extracted from maps.

7. References

Agazzi, O. E., Kuo, S., Levin, E. & Pieraccini, R. (1993). Connected and degraded text recognition using planar hidden Markov models, *Proceedings ICASSP'93*, pp. V113–V116.

Baird, H. S. (1992). Document image defect models, *in* H. S. Baird, H. Bunke & K. Yamamoto (eds), *Structured Document Image Analysis*, Springer, pp. 546–556.

Baird, H. S. (1999). Document image quality: Making fine discriminations, *In Proc. IAPR Int'l Conf. on Document Analysis and Recognition*, pp. 459–462.

Baird, H. S. (2007). The state of the art of document image degradation modelling, *in* B. Chaudhuri (ed.), *Digital Document Processing: Major Directions and Recent Advances*, Springer, pp. 261–279.

Bose, C. B. & Kuo, S. S. (1994). Connected and degraded text recognition using hidden markov model, *Pattern Recognition* 27: 1345–1363.

Cao, R. & Tan, C. L. (2002). Text/graphics separation in maps, *GREC '01: Selected Papers from the Fourth International Workshop on Graphics Recognition Algorithms and Applications*, Springer-Verlag, London, UK, pp. 167–177.

Casey, R. G. & Lecolinet, E. (1996). A survey of methods and strategies in character segmentation, *IEEE Trans. Pattern Anal. Mach. Intell.* 18(7): 690–706.

Chen, L., Liao, H., Wang, J. & Fan, K. (1999). Automatic data capture for geographic information systems, *IEEE Trans. System, Man, and Cybernetics-part C: Applications and Reviews* 29(2): 205–215.

Chiang, Y., Knoblock, C. A. & Chen, C. (2005). Automatic extraction of road intersections from raster maps, *GIS '05: Proceedings of the 13th annual ACM international workshop on Geographic information systems*, pp. 267–276.

Dhar, D. B. & Chanda, B. (2006). Extraction and recognition of geographical features from paper maps, *Int. J. Doc. Anal. Recognit.* 8(4): 232–245.

Dori, D. & Wenyin, L. (1999). Automated CAD conversion with the machine drawing understanding system: Concepts, algorithms, and performance, *IEEE Trans. System, Man, and Cybernetics-part A: System and Humans* 29(4): 411–416.

Elms, A., Procter, S. & Illingworth, J. (1998). The advantage of using an HMM-based approach for faxed word recognition, *Int. J. Doc. Anal. Recognit.* 1(1): 18–36.

Fischler, M. A. & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, *Commun. ACM* 24(6): 381–395.

Gamba, P. & Mecocci, A. (1999). Perceptual grouping for symbol chain tracking in digitized topographic maps, *Pattern Recogn. Lett.* 20(4): 355–365.

Gelbukh, A. F., Levachkine, S. & Han, S.-Y. (2003). Resolving ambiguities in toponym recognition in cartographic maps, *GREC*, pp. 75–86.

Guillevic, D. & Suen, C. (1997). An HMM word recognition engine, *ICDAR'97*, IEEE Computer Society, pp. 544–547.

Hull, J. (1998). Document image skew detection: Survey and annotated bibliography, *in* J. Hull & S. Taylor (eds), *Document Analysis Systems II*, World Scientific, pp. 40–64.

Hull, J., Srihari, S. & Choudhari, R. (1983). An integrated algorithm for text recognition: Comparison with a cascaded algorithm, *IEEE Trans. Pattern Anal. Mach. Intell.* 5(4): 384–395.

Kanungo, T., Haralick, R. M., Baird, H. S., Stuezle, W. & Madigan, D. (2000). A statistical, nonparametric methodology for document degradation model validation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22: 1209–1223.

Kerle, N. & Leeuw, J. (2009). Reviving legacy population maps with object-oriented image processing techniques, *IEEE Transactions on Geoscience and Remote Sensing* 47(7): 2392–2402.

Khotanzad, A. & Zink, E. (2003). Contour line and geographic feature extraction from USGS color topographical paper maps, *IEEE Trans. Pattern Anal. Mach. Intell.* 25(1): 18–31.

Kim, H., Kim, S., Kim, K. & Lee, J. (1997). An HMM-based character-recognition network using level building, *Pattern Recognition* 30(3): 491–502.

Koerich, A., Koerich, R. L., Sabourin, R., El-Yacoubi, A. & Suen, C. Y. (2000). A syntax-directed level building algorithm for large vocabulary handwritten word recognition, *In Proc. 4th International Workshop on Document Analysis Systems*, pp. 255–266.

Konheim, A. G. (1981). *Cryptography: A Primer*, Wiley, New York, pp. 24–25.

Kuo, S. S. & Agazzi, O. E. (1994). Keyword spotting in poorly printed documents using pseudo 2-d hidden Markov models, *IEEE Trans. Pattern Anal. Mach. Intell.* 16(8): 842–848.

Levachkine, S., Velázquez, A., Alexandrov, V. & Kharinov, M. (2002). Semantic analysis and recognition of raster-scanned color cartographic images, *GREC '01: Selected Papers from the Fourth International Workshop on Graphics Recognition Algorithms and Applications*, pp. 178–189.

Levin, E. & Pieraccini, R. (1992). Dynamic planar warping for optical character recognition, *Proceedings ICASSP'92*, pp. III 149–III 152.

Leyk, S., Boesch, R. & Weibel, R. (2006). Saliency and semantic processing: Extracting forest cover from historical topographic maps, *Pattern Recogn.* 39(5): 953–968.

Li, L., Nagy, G., Samal, A., Seth, S. C. & Xu, Y. (2000). Integrated text and line-art extraction from a topographic map, *Int. J. of Doc. Anal. Recognit.* 2(4): 177–185.

Liang, S., Sridhar, M. & Ahmadi, M. (1994). Segmentation of touching characters in printed document recognition, *Pattern Recognition* 27(6): 825–840.

Likforman-Sulem, L. & Sigelle, M. (2009). Combination of dynamic bayesian network classifiers for the recognition of degraded characters, *SPIE Document Recognition and Retrieval XVI*, pp. 1–10.

Lu, Z. (1998). Detection of text regions from digital engineering drawings, *IEEE Trans. Pattern Anal. Mach. Intell.* 20(4): 431–439.

Namane, A., Soubari, E., Djebari, A., Meyruels, P. & Bruynooghe, M. (2006). Hopfield-multilayer-perceptron serial combination for accurate degraded printed character recognition, *Optical Engineering* 45(8): 087201.1–087201.15.

Pezeshk, A. & Tutwiler, R. L. (2008a). Contour line recognition and extraction from scanned color maps using dual quantization of the intensity image, *IEEE Southwest Symposium on Image Analysis and Interpretation*, pp. 173–176.

Pezeshk, A. & Tutwiler, R. L. (2008b). Text segmentation and reorientation from scanned color topographic maps, *10th IASTED Intl. Conference on Signal and Image Processing*, pp. 94–97.

Pezeshk, A. & Tutwiler, R. L. (2010a). Extended character defect model for recognition of text from maps, *IEEE Southwest Symposium on Image Analysis and Interpretation*, pp. 85–88.

Pezeshk, A. & Tutwiler, R. L. (2010b). Improved Multi Angled Parallelism for separation of text from intersecting linear features in scanned topographic maps, *ICASSP 2010*, pp. 1078–1081.

Pouderoux, J., Gonzato, J.-C., Pereira, A. & Guitton, P. (2007). Toponym recognition in scanned color topographic maps, *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition*, pp. 531–535.

Rabiner, L. R. & Levinson, S. E. (1985). A speaker-independent, syntax-directed, connected word recognition system based on hidden Markov models and level building, *IEEE Trans. Acoustics, Speech, and Signal Processing* ASSP-33(3): 561–573.

Roy, P. P., Llados, J. & Pal, U. (2007). Text/graphics separation in color maps, *ICCTA '07: Proceedings of the International Conference on Computing: Theory and Applications*, pp. 545–551.

Sarfraz, M., Zidouri, A. & Shahab, S. (2005). A novel approach for skew detection of document images in OCR system, *CGIV'05*, IEEE Computer Society, pp. 175–180.

Schenkel, M. & Jabri, M. (1998). Low resolution, degraded document recognition using neural networks and hidden Markov models, *Pattern Recognition Letters* 19(3-4): 365–371.

Smith, E. H. B. (2001). Estimating scanning characteristics from corners, *In Proc. SPIE Document Recognition and Retrieval VIII, Volume 4307*, pp. 176–183.

Tombre, K., Tabbone, S., Pélissier, L., Lamiroy, B. & Dosch, P. (2002). Text/graphics separation revisited, *in: Workshop on Document Analysis Systems (DAS)*, Springer-Verlag, pp. 200–211.

Velázquez, A. & Levachkine, S. (2003). Text/graphics separation and recognition in raster-scanned color cartographic maps, *the Fifth International Workshop on Graphics Recognition Algorithms and Applications(GREC2001)*, pp. 63–74.

Wenyin, L., Zhang, W. & Yan, L. (2007). An interactive example-driven approach to graphics recognition in engineering drawings, *Int. J. Doc. Anal. Recognit.* 9(1): 13–29.

Recent Advances in Document Recognition and Understanding

Edited by Dr. Minoru Mori

In the field of document recognition and understanding, whereas scanned paper documents were previously the only recognition target, various new media such as camera-captured documents, videos, and natural scene images have recently started to attract attention because of the growth of the Internet/WWW and the rapid adoption of low-priced digital cameras/videos. The keys to the breakthrough include character detection from complex backgrounds, discrimination of characters from non-characters, modern or ancient unique font recognition, fast retrieval technique from large-scaled scanned documents, multi-lingual OCR, and unconstrained handwriting recognition. This book aims to present recent advances, applications, and new ideas that are relevant to document recognition and understanding, from technical topics such as image processing, feature extraction or classification, to new applications like camera-based recognition or character-based natural scene analysis. The goal of this book is to provide a new trend and a reference source for academic research and for professionals working in the document recognition and understanding field

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Aria Pezeshk and Richard L. Tutwiler (2011). Character Degradation Model and HMM Word Recognition System for Text Extracted from Maps, Recent Advances in Document Recognition and Understanding, Dr. Minoru Mori (Ed.), ISBN: 978-953-307-320-0, InTech, Available from:
http://www.intechopen.com/books/recent-advances-in-document-recognition-and-understanding/character-degradation-model-and-hmm-word-recognition-system-for-text-extracted-from-maps